MEMORIAL UNIVERSITY OF NEWFOUNDLAND

DEPARTMENT OF MATHEMATICS AND STATISTICS

FINAL EXAM **MATH 6205** APRIL 16TH, 2022

Notes (please read carefully!):

- This exam consists of four questions, but only three of them have to be completed. You can choose any three questions that you like, but you clearly have to indicate which problems you have chosen. Only those problems will be marked then.
- The maximum number of points of this exam is 30.
- This exam is open book. You may use any aides of your choice, however, directly copying code from the internet will not be tolerated. Answers that include code taken directly from the internet will receive zero points.
- If you complete both questions 1 and 2, you cannot use the same model/method for both questions.
- This assignment has to be completed by yourself alone.
- Clearly document all your workings, e.g. using code commenting, jupyter's text functionality, etc. If a question asks for "Interpret your results" and this is not done explicitly, you will receive point deductions.
- Upload your solutions to the Brightspace *Final exam* Dropbox by 6:00 pm NDT.

Questions:

1. [**10 points**] Implement an *image restoration* routine in Keras. That is, the goal is to train a suitable neural network to restore corrupted images. Complete the following tasks:

   (a) Download the `ImageReconstruction.ipynb` notebook from Brightspace. This notebook contains a corruption routine which will corrupt pixels in an image dataset. The size and number of the corrupted image blocks can be specified with the `block_size` and `max_no_blocks` arguments. The larger the block size and number of blocks, the more corrupted the images will appear.

   (b) Corrupt the CIFAR-10 dataset with `max_no_blocks` $\in \{10, 50, 100\}$ (for a fixed `block_size`=3), and implement a suitable model that will accept a corrupted image as input and the corresponding original (i.e. complete) image as output.

   (c) Visualize some of the reconstructions of your model applied to the test dataset. Below is an example of how this reconstruction may look:

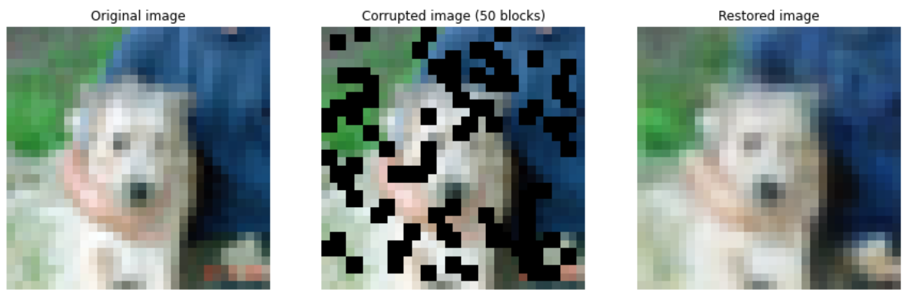


Figure 1: Sample reconstruction.

   (d) Compute the loss and SSIM over the test dataset as a function of the corruption strength and interpret your results.

2. **[10 points]** Implement an *image colorization* routine in Keras. That is, the goal is to train a model to convert a gray-scale image (i.e. an image with a single color channel) to an RGB image (i.e. and image with three color channels). Complete the following tasks:

   (a) Implement a suitable model that will accept a gray-scale image as input and will produce an RGB image as output.

   (b) The dataset to use for this problem is the DIV2K dataset we had seen for our image upscaling tasks. Note that this dataset pairs low-resolution to high-resolution images. Here only use the low-resolution images, resize them to the uniform image size of 64×64 pixels, with the input image being the low-resolution, resized image converted to gray-scale and the output image being the low-resolution, resized RGB image.

   (c) Visualize some of the reconstructions of your model applied to the test dataset. Below is an example of how this reconstruction may look:

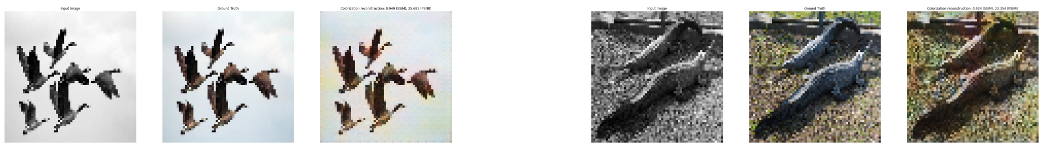

Input Image
Ground Truth
Colorization reconstruction: 0.949 (SSIM), 25.683 (PSNR)
Input Image
Ground Truth
Colorization reconstruction: 0.924 (SSIM), 23.354 (PSNR)

Figure 2: Sample results of an image colorization algorithm.

   (d) Compute the PSNR and SSIM over the test dataset and interpret your results.

3. [**10 points**] Download the `Lorenz96.ipynb` notebook from Brightspace. This notebook provides a numerical solution for the Lorenz–1996 model, which is frequently being used for data assimilation research. This model reads

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \qquad i = 1, \dots, n$$

where $x_{-1} = x_{n-1}$, $x_0 = x_n$ and $x_{n+1} = x_1$ and $n \geq 4$, with F being a constant forcing parameter. Here we use $F = 8$. The solution vector is $\mathbf{x}(t) = (x_1(t), \dots, x_n(t))$.

The notebook provides a numerical solution with initial condition $\mathbf{x}(0) = (8.01, 8, 8, 8, 8)$, over the time interval $[0, 100]$, which consists of 10,000 time steps. The goal is to predict the future evolution of $\mathbf{x}(t)$ given past values of this solution vector.

Complete the following tasks:

(a) Split the computed numerical solution into a training and a test dataset. Use 80% of the data for training, and 20% for testing. The goal is to predict the next 64 values, given the past 128 values, using any suitable method of your choice.

(b) Your model should work for any value of n, so please test this and plot some individual predictions of your model, comparing them to the respective numerical solutions. Note that if you use an n different from $n = 5$ you have to generate a new numerical solution. Simply use $\mathbf{z}(0) = (8.01, 8, \dots, 8)$. As a reference, below are shown some of the predictions that may be possible with a suitable model (here for $n = 5$):

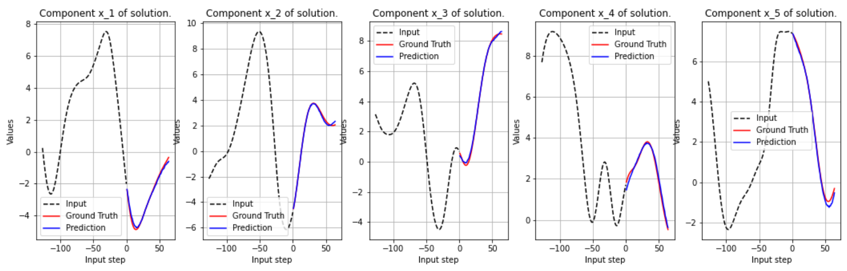


Figure 3: Sample prediction for the Lorenz–1996 model.

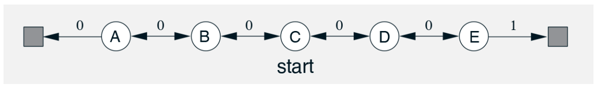
(c) Compute the mean squared error of your model over the testing dataset.

(d) To further assess the generalization abilities of your model, generate a new dataset by producing a new numerical solution for the initial condition

$$\mathbf{x}(0) = (8.65, -2.42, 2.48, 1.04, 3.41),$$

but do *not* retrain your model from before. Again split your dataset into training and testing datasets, and simply discard the training dataset (as we are not retraining the model!). Simply apply your pre-trained model to the new test dataset, plot some predictions and evaluate the mean squared error over the new test dataset. Is your model still working accurately for the new test dataset? Provide some explanation for your observations!

4. **[10 points]** Consider the following Markov reward process:



All episodes start in the center state C and proceed with equal probability to the left or to the right by one state on each step. The episodes terminate either on the very left or on the very right (gray states). All rewards are zero except when the episodes terminates on the right, in which case a reward of $+1$ is issued. For example, one possible state-reward sequence would be: C, 0, B, 0, C, 0, D, 0, E, 1. We do not use discounting, i.e. $\gamma = 1$. Thus, the true value of each state is the probability of terminating on the right if starting from that state, meaning the true values of all states A through E are $\frac{1}{6}$, $\frac{2}{6}$, $\frac{3}{6}$, $\frac{4}{6}$, and $\frac{5}{6}$.

The goal of this question is to empirically compare the prediction abilities of the TD(0) and the constant-α Monte Carlo prediction algorithms when applied to this Markov reward process, i.e. to assess which algorithm converges more quickly.

Complete the following objectives:

(a) Implement both the TD(0) and the every-visit Monte Carlo prediction method for constant α. The value function update formulas for these algorithms can be found in *Part 8 – Reinforcement learning 2* (page 62).

(b) Show the values learned by both algorithms after various numbers of episodes for a single run using $v(s) = 0.5$ as initial estimate for the value of all states. For example, below (left plot) is shown how TD(0) with $\alpha = 0.1$ estimates the values of each state after 0, 1, 10 and 100 episodes. Reproduce this plot (modulo randomness!) and show the same plot for Monte Carlo with $\alpha = 0.03$.

(c) Plot the RMSE between the value function learned and the true value function, averaged over the five states and then averaged over a total of 100 runs as a function of the number of episodes. You should get a plot similar as the one below (right plot).

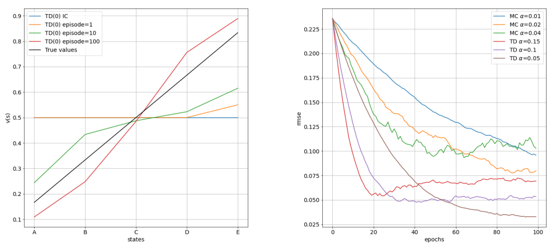


Figure 4: **Left:** Estimated values as obtained from TD(0) using $\alpha = 0.1$ for a number of episodes. **Right:** Empirical RMSE averaged over all states as a function of the number of episodes carried out. Results have been averaged over 100 runs.